System Call In Operating system

Normal function call vs System Calls in OS

Normal Function Call

A normal function call is an instruction that transfers control to a function in the same user space (or within the same program).

- Executed in user mode (does not require kernel intervention).
- Has direct access to process memory.
- Generally faster because there is no mode switch.
- Used for computations, data manipulations, and program logic execution.

Ex : add() is a normal function that executes within the same user space.

```c
#include <stdio.h>

void add(int a, int b) {
    printf("Sum: %d\n", a + b);
}

int main() {
    add(5, 10);  // Normal function call
    return 0;
}
```

System Call In Operating system

```
#include <unistd.h>

int main() {
    char msg[] = "Hello, System Call!\n";
    write(1, msg, sizeof(msg));  // System Call: Writes to sta
    return 0;
}
```

Normal function call vs System Calls in OS

System Call

A system call is a request made by a user-space program to access privileged OS services, such as file handling, process management, or memory allocation.

- Switches from user mode to kernel mode (context switch required).
- Executed in kernel mode (higher privilege level).
- Used to interact with hardware, manage processes, or handle files.
- Slower than normal function calls due to mode switching.

Ex : write() is a system call that interacts with the OS to write data to the terminal.

System Call In Operating system

A **System call** is like a request your program makes to the operating system (OS) when it needs to do something important, like:
-> Opening a file
-> Creating or closing a program
-> Talking to another program
-> Using hardware (like a printer or keyboard)
Your program cannot directly control the computer's hardware, so it asks the OS for help using system calls.

Types of System Calls

1. Process Control
2. File Management
3. Device Management
4. Information Maintenance
5. Communication (IPC - Inter-Process Communication)

Normal function-> when working within the program's scope without needing OS intervention.

System calls -> when you need to interact with the operating system, like handling files, processes, or memory.

System Call In Operating system

1. Process Control: System calls for creating, terminating, or managing processes.

- fork() – Creates a new process.
- exec() – Replaces the current process with a new program.
- exit() – Terminates a process.
- wait() – Waits for a child process to finish execution.
- kill() – Sends a signal to terminate a process.
- getpid() – Gets the process ID.

2. File Management: System calls for handling files.

- open() – Opens a file.
- read() – Reads from a file.
- write() – Writes to a file.
- close() – Closes a file.
- lseek() – Moves the file pointer.
- unlink() – Deletes a file.

System Call In Operating system

3. Device Management: Used to interact with hardware devices.

- ioctl() – Sends a control command to a device.
- read() – Reads from a device.
- write() – Writes to a device.

4. Information Maintenance: Retrieves system-related information.

- getpid() – Gets the process ID.
- getuid() – Gets the user ID.
- alarm() – Sets a timer for process execution.

System Call In Operating system

5. Communication (IPC - Inter-Process Communication) : System calls for processes to communicate with each other.

- pipe() – Creates a communication channel between processes.
- shmget() – Creates shared memory.
- msgsnd() – Sends a message to a queue.
- msgrcv() – Receives a message from a queue.
- socket() – Creates a network socket.
- send(), recv() – Send and receive messages over a socket.